CSN-341
Computer Networks
Mini-Project: Interim Evaluation Report

Group-6

Name	Enrollment No
Meet Sindhav	22114053
Mohammed Haaziq	22114055
Aditya Mundada	22114058
Nayan Kakade	22114060
Sarvesh Baheti	22114087
Roopam Taneja	22125030

Topic: Network Management Tool with SNMP

Abstract

We are developing an SNMP Tool which will be able to monitor metrics related to the client like CPU usage, memory usage, Wi-Fi incoming packets, etc. The client will also have the ability to send traps (real-time alerts) to the server regarding its current situation, such as if its system is rebooting, experiencing a cold start, etc. We will visualize the results using the data received from the client by plotting graphs with time and the corresponding metrics.

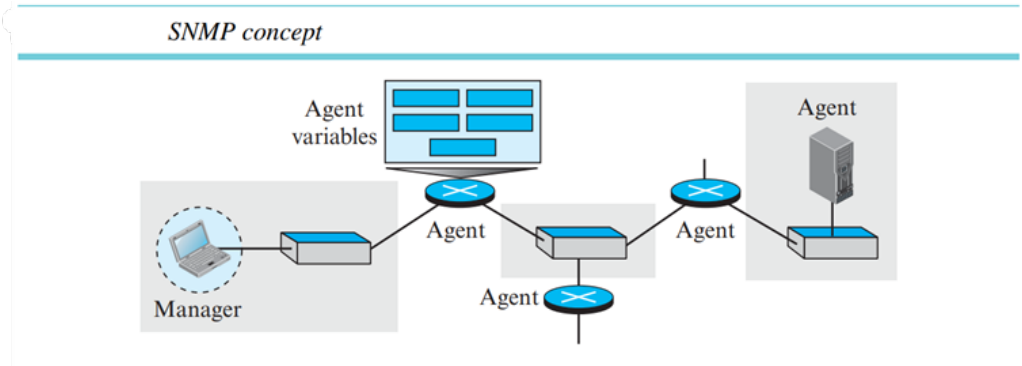
Theory & Approach

SNMP is an application-level protocol in which a few manager stations control a set of agents. The protocol is designed at the application level so that it can monitor devices made by different manufacturers and installed on different physical networks. In other words, SNMP frees management tasks from both the physical characteristics of the managed devices and the underlying networking technology. It can be used in a heterogeneous internet made of different LANs and WANs connected by routers made by different manufacturers. To perform management tasks, SNMP uses two other protocols: the Structure of Management Information (SMI) and the Management Information Base (MIB).

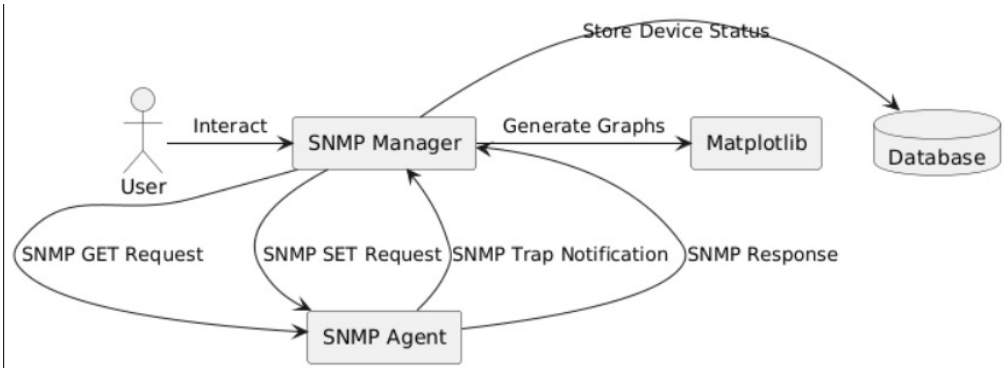
Our approach relies on the APIs provided by the Python `pysnmp` library such as `snmpget`, `snmpset`, and `snmptrap`. We also need to configure our system's SNMP files to allow sending and receiving of SNMP commands. This is easier done in the Ubuntu Linux environment, as Linux allows more flexibility than Windows.

For tracking the client's device status and fetching network details through SNMP, we use asynchronous functions to send SNMP requests to the device. We will collect OIDs like system description, CPU usage, memory, and network packets, and send an SNMP GET request. Using this, we will print the retrieved information by matching the OIDs with their names. We will also provide error handling.

Using the data received, we will be able to plot graphs like scatterplots, bar graphs, etc., preferably using the `matplotlib` library.



SNMP notifications provide a way for an SNMP agent to send an asynchronous notification about conditions that the SNMP manager(s) might care about. Traps are typically produced by the SNMP agent. To receive traps, we set up an SNMP engine to listen to specific IP addresses of registered clients. It will use the community string, which is present in the `snmpd.conf` file for authentication, and will log and display the OIDs and values upon receiving a trap. The engine will continuously listen for traps until manually stopped.



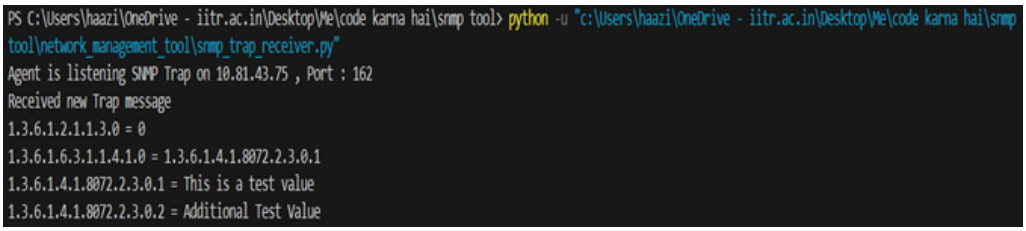
UML Diagram

Current Progress

We were able to monitor the client's network and system following the algorithm described above.

```
[Running] python -u "c:\Users\Asus\OneDrive\Desktop\Projects\Acad\snmp\snmp_manager.py"
Device 10.81.66.166 info:
System Description: Linux nayan-Inspiron-3501 6.8.0-45-generic #45~22.04.1-Ubuntu SMP
PREEMPT_DYNAMIC Wed Sep 11 15:25:05 UTC 2 x86_64
System Location: Sitting on the Dock of the Bay
Wifi incoming packets: 9486475
Ethernet incoming packets: 0
Wifi outgoing packets: 2065025
Ethernet outgoing packets: 0
Wifi inbound errors: 0
Ethernet inbound errors: 0
Wifi outbound errors: 0
Ethernet outbound errors: 0
System Uptime: 56170
Number of Processes: 305
CPU 1 Utilisation: 3
CPU 2 Utilisation: 3
Total Memory Size: 7848888
Memory used: 4205032
```

We were also able to receive alerts from the client.



Now, all that is left is to plot the data received to visualize the trends in the client's system and network performance. If time permits, we will add UI/UX design to our tool and also improve security by using SNMPv3 to provide encryption and try to limit the number of traps that a client can send at a time to avod attacks like denial-of-service (DoS) attacks, where a few clients send too many traps to the server.

What Problem Does It Solve?

- **Real-Time Monitoring:** This project will allow admins to monitor device status (like routers, switches, servers), helping to ensure network stability by tracking critical metrics like CPU usage, memory, bandwidth, and packet errors.
- **Early Issue Detection:** With real-time alerts and SNMP traps, it notifies admins about potential issues (like high CPU usage, device failures, or traffic spikes).
- **Performance Analysis:** The tool's performance graphs visualize historical and real-time data, allowing network administrators to identify trends, bottlenecks, or inefficiencies, leading to proactive maintenance and optimization.